# Abstract

Text mining refers to the process of transforming unstructured text data into structured clusters of information. This project explores a specific aspect of text mining known as authorship attribution, which involves analyzing various linguistic and stylistic features of text to predict its author. This project assesses transcription data containing the speeches delivered by South African presidents during the State of the Nation Address (SONA) from 1994 to 2023. The primary goal was to develop a classification model that can take a sentence from a SONA speech as input and correctly predict the president who said it. Various models were evaluated in this project, such as Classification Trees, and Random Forests, in addition to XGBoost  - , Naïve Bayesian- and feed forward Neural Network models. However, I find that …. outperformed all other models with a test ….F1-score of…

Introduction

Text mining is a branch of artificial intelligence (AI) that aims to transform unstructured text data into structured formats (Ibm.com, 2023)1. Text mining employs a variety of statistical and machine learning methods, including deep learning algorithms. These methods are used to uncover textual patterns, trends, and hidden relationships within unstructured data.

While traditional text mining relied solely on machine learning algorithms, modern text mining also employs sophisticated methods of Natural Language Processing (NLP) such as parsing and part-of-speech tagging (Greenbook.org, 2017)2. This advancement in text mining is largely attributed to the exponential growth in data, with approximately 80% of global data residing in unstructured formats. This vast amount of data has necessitated the use of text mining, making it a significant task in data analytics.

Text mining is particularly useful in large organizations where decision-making is central and time is limiting (Ibm.com, 2023)1. For example, banks employ text mining in risk management to scrutinize changes in the sentiment of financial reports, which may provide insight on potential investments.

The many applications of text mining across various domains has led to the development of several models, including supervised, and unsupervised methods (Dogra et al., 2022)3. However, determining the most appropriate and effective model for a specific text mining task remains a complex and nuanced challenge.

The aim of this project is to identify the most effective classification model for authorship attribution, which involves determining the author of a given document (Mohamed Amine Boukhaled and Jean-Gabriel Ganascia, 2017)4. This project specifically focuses on analyzing transcription text data from speeches delivered by South African presidents during the State of the Nation Address (SONA) from 1994 to 2023 (www.gov.za, 2023)5. SONA serves as an annual opening to South African Parliament, where the President reports on the socio-economic state of the nation to a joint sitting of Parliament. The main objective is to train a model that can take a sentence from a SONA speech and correctly classify which president said it.

# Literature Review

Several studies have machine learning methods for author prediction. These techniques include Support Vector Machines (SVMs), Decision Trees, Random Forests (RF), and Neural Networks (NNs), to name a few. The choice of model often depends on the nature of the data and the specific requirements of the task.

Feature selection plays a crucial role in author prediction, where features are broadly categorized into lexical features (e.g., word usage) and syntactic features (e.g., part-of-speech tags). Some studies have also explored semantic features (e.g., topics and sentiments).

An article by Shukri, (2021)6 suggests a method for author prediction, by training models on Arabic opinion articles. The study collected 8109 articles from 428 authors for the period 2016 to 2021. Their NN model achieved the highest accuracy of 81.1%.

Another article by Bauersfeld et al., (2023)7 proposed a transformer-based, neural-network architecture that uses text content and author names in the bibliography to determine the author of an anonymous manuscript. The authors used all research papers publicly available on arXiv and achieved a 73% accuracy rate.

A similar paper by Khalid, (2021)8 performed author prediction on 210 000 anonymous, news headlines from HuffPost (2012-2022). The study used Bag of Words (BoW) and Latent Semantic Analysis (LSA) features as input to train classification algorithms such as LR and RF. The study found that the LR model trained on all features outperformed all other models with an accuracy of 94.9%.

These manuscripts demonstrate the potential of text mining in various applications. These papers also highlight the variety of author-prediction techniques available, such as NN-, LR-, and RF- and SVM- models. However, we also note the challenges in applying these techniques, such as the need for large datasets, as well as the ability of these models to discriminate between content-related features and author-specific features.

# Data

## Data Source

The SONA dataset is publicly available on the South African government website (www.gov.za, 2023)5. The data contains the speeches delivered by South African presidents at the annual State of the Nation Address (SONA) from 1994 to 2023.

Seven speeches are available for former president Mandela (1994-1999). Ten speeches are available for former president Mbeki (2000-2008). Ten speeches are also available for former president Zuma (2009-2017). The current president (Ramaphosa) has a total of seven speeches to date (2018-2023). Two outliers exist, namely one speech for former president deKlerk (1994) and former president Motlanthe (2009).

These records cumulatively formed the SONA dataset with 36 records and 5 variables, namely: filename, speech, year, president and date of speech delivered.

## Data Pre-processing

All data was read into R version 4.3.1 , using R-Studio version 2023.9.1.494. The year of each speech was extracted from the first four characters in the “filename” column. The president names were extracted from the “filename” column using regular expressions, where alphabetical text ending in a “.txt” extension was matched as the presidents’ name. Subsequently, all other unnecessary text such as “http”-, fullstop-, ampersand-, greater-than-, and less-than characters were removed, in addition to trailing white spaces and new-line characters. Dates were then re-formatted into a *dd-mm-yyyy* format. Finally, the pre-processed data was saved as an RDS object for downstream analysis.

Methods

## Data processing

The pre-processed data was read into R and converted to tibble format. The data was then assessed by looking at the head and tail of the tibble, in addition to looking at the data types of each column. Dates which appeared before a president’s speech were removed using regular expressions. Trailing white spaces before and after a president’s speech were also removed. I also noticed that former president- Motlanthe and deKlerk only had one record in comparison to the other presidents with more than one record, and so these observations were removed from the data set.

As a result, the processed data had 34 rows and 5 columns.

## Tokenization

The processed SONA dataset was then tokenized into sentences using unnest\_tokens(), since the goal is to make predictions on sentence inputs. All text was then converted to lowercase to remove word redundancy, where each word is treated as its own unique feature, irrespective of letter case. I also removed all punctuation so that root-words are treated alike. I then included a sentence ID column to track sentence membership.

The sentence tokens were then tokenized into word and bigram tokens respectively, where all stop words were removed. I also ensured that "blank" tokens were removed. For the tokenization by bigram implementation, bigrams were first split into individual words, where each word was assessed for stop words. If a stop word was detected, the entire bigram was removed, while the remaining bigrams were unified.

## Exploratory Data Analysis

I looked at the 20 most frequently used- words and bigrams: (1) for all presidents, and (2) for each president. I also looked at the average number of sentences per speech and the average number of words per sentence for each president.

## Features

### Bag-of-Words Model

The Bag-of-Words (BoW) model computes the frequency of occurrence of an unordered collection of words within a document and uses these frequencies as features to train the classifier.

A word bag was generated by taking the word tokens and grouping the unique combinations of "sentence ID, president and word" and computing each grouping's frequency, after which the top 200 words for each grouping was selected to create the final word bag. The choice of the top 200 words was chosen due to its superior model performance relative to the top 100 and 500 word models (tested informally). The word bag consisted of 363 rows (words) and 3 columns, namely, sentence ID, president, and word.

The BoW table was then constructed by identifying all the words in a sentence that overlap with the word bag. The frequency of each word within a sentence was then calculated. Finally, the BoW table was reformatted to a wide format (tidy format), where columns represent the features (words), the rows represent the observations (sentence ID) and the cell values are the frequency of a word within a sentence. All words not found in a sentence obtained a value of 0.

### Term Frequency-Inverse Document Frequency Model

Term Frequency – Inverse Document Frequency (TF-IDF) refers to the metric that describes how important a word is in a document relative to other documents in the corpus. TF-IDF is calculated by multiplying the Term Frequency (TF) by the Inverse Document Frequency (IDF), where TF is the frequency of a word within a document divided by the total number of words in that document, whereas IDF is the proportion of documents in the corpus that contain the word.

Bind\_tf\_idf() was used to calculate the TF-IDF for each word in each sentence. It is important to note that the "document" here refers to the sentence ID. The BoW table previously discussed was then manipulated to include the TF-IDF value instead of the word frequency.

## Class imbalance and up-sampling

I checked for class imbalance by comparing the frequency of records in each of the target variable classes, namely: Mandela, Mbeki, Zuma and Ramaphosa. I found that the classes were imbalanced, where Mbeki had the largest proportion of sentences (92), followed by Ramaphosa (51), Zuma (41) and Mandela (21). As a result, I used upSample() in the Caret package to oversample the minority classes so that the number of observations in each class match that of the majority class.

## Split Balanced Data into Training, Validation and Test sets

I partitioned the data into 70% training and 30% test sets using createDataPartition(), which performs stratified sampling, ensuring balanced classes in each split. The target variable, president was then converted to factors, where classes: Mandela, Mbeki, Ramaphosa and Zuma were categorized as levels 1 to 4, respectively. For the validation set, 5-fold cross-validation was applied during training to determine the optimal model parameters by means of a grid-search (discussed below).

## Models

Each classification model implemented both the BoW and TF-IDF features discussed above. For each model, a grid search was performed to find the optimal hyperparameters, generating sub-models as a result. For each sub-model, 5-fold cross-validation was performed on the training set. The best sub-model was determined by the model performance on both the training and validation sets. The hyperparameters of the best sub-model was then used to re-train the model on the full training set, after which predictions were made on the test set. The final models were compared based on their test set performance. **Figure 1** below shows the general workflow.

**Figure 1:** Workflow of model building after splitting data into 70 % training and 30 % test sets

### Classification Tree

Classification trees are decision trees that recursively partition the input space and assign a class label to each partitioned region based on the majority class of the training samples in that region. A grid search was performed using the rpart() function with the following parameters:

* The complexity parameter, a stopping criterion where tree splitting terminates once the reduction in relative error is less than a specified cp-threshold.

**cp** = {0.001, 0.112, 0.223, 0.334, 0.445, 0.556, 0.667, 0.778, 0.889, 1.000}

* The minimum number of observations at any terminal node.

**minbucket** = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

* The minimum number of observations that must exist in a node for a split to be attempted.

**minsplit** = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

### Random Forest

The Random Forest algorithm uses bagging to train multiple decision trees in parallel, reaching a majority vote of class classification. This type of ensemble learning helps to overcome overfitting.

The randomForest() function was used, taking in the following parameters:

* The number of trees to grow.

**ntree** = {100, 500, 1000}

* Feature importance.

**Importance** = TRUE

* **na.action** = na.exclude

### Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) models aggregate the output of a sequential ensemble of tree models, where each subsequent model improves on the previous model. The xgboost() function was used, with the following important parameters:

* The maximum depth of the tree, where increasing this value results in a more complex model that is likely to overfit.

**max depth** = {1, 2, 3, 4, 5, 6}

* The step size shrinkage, which is used to prevent overfitting. This value of eta shrinks the feature weights to make the boosting process more conservative.

**eta** = {0.2, 0.4, 0.6, 1}

* The minimum loss reduction, where the larger the value of gamma is, the more conservative the algorithm is.

**gamma** = {0.5, 1}

* The L2 regularization term on the weights, where the default lambda value of 1 was chosen.

**lambda** = 1

* **objective** = “multi:softmax” objective function for multi-class classification.

It is important to note, that the labels of the input data was converted to factors then to integers and finally offset by -1 because xgboost() takes in numeric data where classes of the target variable should be indexed from 0. The features need to be converted to a DMatrix object using the xgb.DMatrix() function. Consequently, I had to offset the predictions by +1 when computing the confusion matrix, since R indexes from 1. I then had to convert the numeric predictions back to factors, where:

* 0+1: Mandela
* 1+1: Mbeki
* 2+1: Ramaphosa
* 3+1: Zuma

### Naïve Bayes

Naïve Bayes is a generative model that seeks to model the distribution of inputs of a given class or category. The naiveBayes() function was used, while applying Laplace smoothing to handle zero probabilities in categorical data. Laplace values of 0, 0.1 and 1 were investigated.

### Feed Forward Neural Network

A feed-forward neural network is a type of artificial neural network where the connections between nodes do not form a cycle. Information in this network moves only in one direction, forward, from the input layer through the hidden layers, to the output layer.

The Keras package in conjunction with Tensorflow was used to construct the feed forward neural network in a Python virtual environment within R-studio, by using the Anaconda software to create the virtual environment. The target variable (president) was extracted from the training data, and converted to factors, after which the factors were converted to discrete integers from 0 to 3, where:

* 0: Mandela
* 1: Mbeki
* 2: Ramaphosa
* 3: Zuma.

Subsequently, the target variable president was one hot-coded to a binary representation. The features were then extracted and converted to a matrix format. Subsequently, a grid search was performed, whereby 5-fold cross validation was applied to each sub-model with the following hyperparameters:

* **Input neurons =** {50, 100, 200}
* **Learning rate =** {0.001, 0.01, 0.1}
* **Drop-out rate =** {0.01, 0.1}

The network was built as follows:

* **Layer 1:** a dense fully connect layer with x input neurons specified above, and which uses either a hyperbolic tangent (Tanh) or ReLU activation function. This layer expects 149 predictor variables (words).
* **Layer 2:** a dropout layer to prevent overfitting by randomly selecting a fraction (specified by the drop-out rate above) of the input units at each update and setting them to 0 during training time.
* **Layer 3:** a dense fully connected layer with 4 output neurons (representing the 4 president classes). A softmax activation function was used to generate a vector of probabilities of class membership for each observation.

The categorical cross-entropy loss function was minimized which is invariant to shifting of the predicted probabilities. The n-Adam optimizer was used to update the model parameters (weights and biases) and to minimize the loss function. This choice of optimizer was based on its superior performance compared to other optimizers, such as SGD, RMSprop, and Adam (informally tested). Finally, the data was trained for 30 epochs, with a batch size of 5, while shuffling the training data at each epoch.

## Performance metrics

1. or all models, performance on the training, validation and full training and test sets were determined by the metrics discussed below. For 5-fold cross-validation, the best sub-model was determined which had the best validation and training macro-average F1-score. The final model was chosen based on the model with the best test set macro-average F1-score. Other metrics were also considered (see below).
2. When training a model with 5-fold cross-validation, each metric is computed at each fold for the training and validation set. At each fold, the metric is computed for every class, and the macro-average metric is obtained by averaging these class-specific metrics (**Equation 1**). After training is complete, the final metric is computed as the average of the macro-average metrics over all folds (**Equation 2**). The macro-average metric was calculated when evaluating model performance on the test set.
3. $$ \text{MacroAverageMetric}\_{\text{j}} = \frac{1}{n} \sum\_{i=1}^{n} \text{metric}\_{i,\text{j}} $$
4. …equation 1, where metrici, j is the metric for class i at a specific fold j , and n is the total number of classes. We see that j = {1, 2, 3, 4, 5}, n = 4 and i = {1, 2, 3, 4}.
5. $$ \text{FinalMetric} = \frac{1}{k} \sum\_{j=1}^{k} \text{MacroAverageMetric}\_{j} $$
6. …equation 2, where MacroAverageMetricj is the macro-average metric at fold j, and k is the total number of folds. We see that k = 5 and j = {1, 2, 3, 4, 5}.

#### Accuracy

1. This is the proportion of correct classifications among the total number of classifications. Accuracy is calculated by taking the sum of the diagonal of the confusion matrix.
2. $$ \text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} $$
3. …equation 3, where TP, TN, FP and FN are the number of true positives, true negatives, false positives, and false negatives, respectively, found in the confusion matrix.

#### Recall

1. Also known as the sensitivity or true positive rate, this is the proportion of actual positives that are correctly classified.
2. $$ \text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} $$…equation 4

#### Precision

1. Also known as the positive predictive value, this is the proportion of positive predictions that are correctly classified.
2. $$ \text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} $$…equation 5

#### F1-score

1. I chose the F1-score as my main criteria for model performance because the F1-score provides information on the model's ability to capture positive cases (recall) and be accurate with the cases it does capture (precision).
2. $$ F1 = 2 \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} $$…equation 6

# Results & Discussion

## Exploratory Data Analysis

#### Top 20 most frequently used words and bigrams

1. **Figure 2** below shows **(a)** the top 20 most frequently used words and **(b)** the top 20 most frequently used bigrams among all presidents. The bigrams provide more context relative to the unigrams, and we see that most president's agenda is about the economy with bigrams such as "economic growth", "job creation" and "economic empowerment". **Figure 3** below shows the top 20 most frequently used words- and **Figure 4** below shows the top 20 most frequently used bigrams- for each president. In these figures we see that the agenda is specific to a president or period of presidency, where Mandela has bigrams such as "people-centered society" alluding to the end of Apartheid, Mbeki focuses on social justice with bigrams such as "social security" and "social partners", Zuma mentions the "world cup" which alludes to the 2010 soccer world cup hosted in South Africa and Ramaphosa's focus is on crime prevention with bigrams such as "gender-based violence" and "law enforcement".

### Average speech and Sentence Length

**Figure 5** below shows the **(a)** average speech length per president and **(b)** the average sentence length per president. In **Figure 5 (a)** we see that Ramaphosa has a relatively long average speech length, in other words the average number of sentences within a speech is quite long for Ramaphosa (327), followed by Zuma (266), Mbeki (242) and Mandela (238). In **Figure 5 (b)** we see that Mbeki (30) has the largest average sentence length (average number of words per sentence), followed by Mandela (25) , Ramaphosa (22) and Zuma (19).

## Bag-of-Word Models

### Classification Trees

In **Table 1** below, we see the results of the best performing classification trees for the BoW features, arranged in descending order of training and mean cross-validation (CV) F1-score.

We see that sub-model 1 and 2 both have the best mean CV F1-score of 0.5462 and a training F1-score of 0.7490. In parallel, we see that that the mean CV and training- accuracy, recall and precision are also the best among all other sub-models. Notably, the training performance is better than the validation performance, where we see very good train accuracy (0.7510) but a small validation accuracy of 0.5652, suggestive of model overfitting. Model 1 parameters were chosen to re-train the full training set.

As a result, we see two interesting clusters in the tree diagram in **Figure i** of **Addendum A**, where Ramaphosa (2018-present) is more likley to talk about health, which is in accordance with the Covid-19 pandemic. In addition, Mandela (1994-1999) is more likely to talk about peace, which is in accordance with the end of the Apartheid era (1994).

Random forests

In **Table 2** below we see that the grid-search performed equally well for all ntree values. We see an improvement in the mean CV-F1 score (0.6476) relative to the BoW classification tree, however, the training F1-score deteriorated (0.6414). All other metrics are relatively good, with values above 0.6, suggesting good model fit. We also see that the training and validation metrics are similar, e.g., training accuracy of 0.6394 is similar to the mean CV accuracy of 0.65. I chose RF sub-model 2 with ntrees = 500 to re-train the full training set, which strikes a balance between too many and too few trees.

### Extreme Gradient Boosting

**Table 3** below suggests that sub-model 47 has the best model performance, and so the model was re-trained on the full training dataset using max depth = 6, eta = 1 and gamma = 0.5. We see that the training F1-score (0.6212) and training accuracy (0.6202) is relatively good, however the mean CV F1-score (0.4256) and mean CV accuracy (0.4384) is relatively low . We see a similar trend for all other metrics, where the training performance is good, but the validation performance is poor, suggestive of model overfitting.

### Naïve Bayes

As seen in **Table 4** below, all "top" performing models displayed poor performance. These models all displayed the same metric values, with a training F1-score of 0.1058 and a mean CV F1-score of 0.1. The training accuracy of 0.25 and mean CV accuracy of 0.2528 also suggests poor model performance. I used the simplest model, model 1 with no Laplace smoothing to re-train the model on the full training dataset.

NB

As seen in **Table 4** below, all “top” performing models displayed poor performance. These models all displayed the same metric values, with a training F1-score of 0.1058 and a mean CV F1-score of 0.1. The training accuracy of 0.25 and mean CV accuracy of 0.2528 also suggests poor model performance. I used the simplest model, model 1 with no Laplace smoothing to re-train the model on the full training dataset.

### Feed Forward Neural Network

In **Table 5** below, we see the results of the feed forward neural network, where model 55 had the best training F1-score of 0.857 (as well as model 17) and the best mean CV F1-score of 0.593. We also see that model 55 had one of the best training accuracies (0.852) and the best validation accuracy (0.612). All other metrics for model 55 are also among the top performing models.

**Figure ii** in **Addendum A** shows the change in the Log Loss, training- recall, precision and accuracy over all 30 epochs. We see that the log loss approaches 0, as the number of epochs increase, whereas the training- recall, precision and accuracy increases.

# Conclusion

In this project, various classification models such as classification trees, random forests, extreme gradient boosting, and neural networks were explored with the goal of predicting which president delivered a specific sentence extracted from a speech. Additionally, both Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) methods were investigated to identify the most frequently used words for each president., where the frequency of the “most frequently” used words are also subjective. This implementation used the top efw words. While different splits of training, test, and validation sets can be considered, this implementation opted for a 70% training and 30% test split. Additionally, a 5-fold cross-validation was employed for the validation set. While both bigram and word tokens were considered for training the classification models, word tokens were ultimately chosen due to their superior performance relative to the bigram tokens. Notably, the sffvs model using ….. outperformed all other models based on the knkl performance metric. In conclusion, this project emphasizes the wide variety of methods available in text mining and further highlights that a “one-approach-fits-all” strategy is not always suitable. One needs to always consider the context, main objectives, and data availability, when deciding on the optimal text mining approach.